Temporal logics for multi-agent systems

Nicolas Markey
LSV, CNRS & ENS Cachan, France

Journées Nationales
Lyon, 21-22 January 2013
Verification of computerised systems

- Computers are everywhere
Verification of computerised systems

- Computers are everywhere

- Bugs are everywhere...

New: Toyota to recall Prius hybrids over ABS software

IDG News Service - Toyota plans to recall around 400,000 of its Prius hybrid cars to replace software that controls the antilock braking system (ABS), the auto maker said Tuesday.
Verification of computerised systems

- Computers are everywhere

- Bugs are everywhere...

- Verification should be everywhere!
Model checking and synthesis

system:

property:

\[ AG(\neg B.\text{overfull} \land \neg B.\text{dried\_up}) \]
Model checking and synthesis

system:

property:

\[ \text{AG}(\neg B.\text{overfull} \land \neg B.\text{dried\_up}) \]
Outline of the presentation

1 Introduction
   → formal verification model checking and synthesis

2 Classical temporal logics: CTL and LTL
   → expressing properties of “closed” systems

3 Temporal logics for games: ATL
   → expressing properties of interacting systems
      extensions to non-zero-sum games
Outline of the presentation

1. Introduction
   - formal verification, model checking, and synthesis

2. Classical temporal logics: CTL and LTL
   - expressing properties of “closed” systems

3. Temporal logics for games: ATL
   - expressing properties of interacting systems
   - extensions to non-zero-sum games
Computation-Tree Logic (CTL*)

- atomic propositions: 0, 0, ...

- boolean combinators: ¬ϕ, ϕ ∨ ψ, ϕ ∧ ψ, ...

- path quantifiers: Eϕ, Aϕ

- temporal modalities: Xϕ, "next ϕ", ϕ U ψ, "ϕ until ψ", ϕ "eventually true U ϕ ≡ F¬¬ϕ ≡ Gϕ" "always ϕ"
Computation-Tree Logic (CTL*)

- atomic propositions: $\bigcirc$, $\bigcirc$, ...
- boolean combinators: $\neg \varphi$, $\varphi \lor \psi$, $\varphi \land \psi$, ...

Computation-Tree Logic (CTL*)

- atomic propositions: \( \Box \), \( \Diamond \), ...
- boolean combinators: \( \neg \varphi \), \( \varphi \lor \psi \), \( \varphi \land \psi \), ...
- path quantifiers:

\[
\begin{align*}
\E \varphi & \quad \varphi \\
\A \varphi & \quad \varphi
\end{align*}
\]
Computation-Tree Logic (CTL*)

- atomic propositions: \( \bigcirc, \bigcirc, \ldots \)
- boolean combinators: \( \neg \varphi, \varphi \lor \psi, \varphi \land \psi, \ldots \)
- path quantifiers:
  - \( E \varphi \)
  - \( A \varphi \)

- temporal modalities:
  - \( X \varphi \)  
    - “next \( \varphi \)”
  - \( \varphi U \psi \)  
    - “\( \varphi \) until \( \psi \)”
Computation-Tree Logic (CTL*)

- **atomic propositions:** \(\circ, \circ, \ldots\)
- **boolean combinators:** \(\neg \varphi, \varphi \lor \psi, \varphi \land \psi, \ldots\)
- **path quantifiers:**
  - \(E\varphi\)
  - \(A\varphi\)
- **temporal modalities:**
  - \(X\varphi\) \(\equiv\) \(\text{“next } \varphi\)"
  - \(\varphi \mathcal{U} \psi\) \(\equiv\) \(\text{“} \varphi \text{ until } \psi\)"
  - \(\text{true } \mathcal{U} \varphi \equiv F\varphi\) \(\equiv\) \(\text{“eventually } \varphi\)"
  - \(\neg F \neg \varphi \equiv G \varphi\) \(\equiv\) \(\text{“always } \varphi\)"
Fragments of CTL*

- CTL: each temporal modality is in the immediate scope of a path quantifier.
Fragments of CTL*

- CTL: each temporal modality is in the immediate scope of a path quantifier.

\[ EF \circ \quad \circ \text{ is reachable} \]
Fragments of CTL*

- CTL: each temporal modality is in the immediate scope of a path quantifier.

\[ \mathbf{E} \Phi \text{ is reachable} \]
Fragments of CTL*

- CTL: each temporal modality is in the immediate scope of a path quantifier.

\[ \text{E G (E F )} \] there is a path along which \( \bigcirc \) is always reachable
Fragments of CTL*  

- CTL: each temporal modality is in the immediate scope of a path quantifier.

\[ \text{E G (E F} \text{)} \]  

there is a path along which \( \vphantom{p} \) is always reachable
Fragments of CTL* 

- **CTL**: each temporal modality is in the immediate scope of a path quantifier.

\[ \text{E} \, \text{G}(\text{E} \, \text{F} \, \text{p}) \quad \text{there is a path along which} \quad \text{p} \quad \text{is always reachable} \]
Fragments of CTL*

- CTL: each temporal modality is in the immediate scope of a path quantifier.

\[
\neg E(\neg \bigcirc) \ U \ \bigcirc \quad \text{in order to reach } \bigcirc, \text{ we have to visit } \bigcirc
\]
Fragments of CTL* 

- CTL: each temporal modality is in the immediate scope of a path quantifier.

$$\neg E(\neg \text{○}) U \text{○}$$ in order to reach ○, we have to visit ○
Fragments of CTL

- CTL: each temporal modality is in the immediate scope of a path quantifier.

Theorem

*CTL model checking is PTIME-complete.*

*CTL symbolic model checking is PSPACE-complete.*
Fragments of CTL

- CTL: each temporal modality is in the immediate scope of a path quantifier.

**Theorem**

*CTL model checking is PTIME-complete.*

*CTL symbolic model checking is PSPACE-complete.*

- LTL: $E\varphi$ or $A\varphi$, where $\varphi$ has no path quantifier.
Fragments of CTL*

- CTL: each temporal modality is in the immediate scope of a path quantifier.

**Theorem**

*CTL model checking is PTIME-complete.*

*CTL symbolic model checking is PSPACE-complete.*

- LTL: \( E\varphi \) or \( A\varphi \), where \( \varphi \) has no path quantifier.

\[ E(G F \bigcirc) \text{ there is a path visiting } \bigcirc \text{ infinitely many times} \]
Fragments of CTL*

- CTL: each temporal modality is in the immediate scope of a path quantifier.

**Theorem**

*CTL model checking is PTIME-complete.*

*CTL symbolic model checking is PSPACE-complete.*

- LTL: $E\varphi$ or $A\varphi$, where $\varphi$ has no path quantifier.

$LTL: E\varphi$ or $A\varphi$, where $\varphi$ has no path quantifier.

$LTL: E(G F \bigcirc)$ there is a path visiting $\bigcirc$ infinitely many times
Fragments of CTL*

- CTL: each temporal modality is in the immediate scope of a path quantifier.

**Theorem**

**CTL model checking** is PTIME-complete.

**CTL symbolic model checking** is PSPACE-complete.

- LTL: \( E\varphi \) or \( A\varphi \), where \( \varphi \) has no path quantifier.

\[
A[(F \bigcirc) \Rightarrow (F G \neg \bigcirc)]
\]

any path that visits \( \bigcirc \) visits \( \bigcirc \) finitely many times
Fragments of CTL

- CTL: each temporal modality is in the immediate scope of a path quantifier.

**Theorem**

CTL model checking is PTIME-complete.

CTL symbolic model checking is PSPACE-complete.

- LTL: $E\varphi$ or $A\varphi$, where $\varphi$ has no path quantifier.

$$A[(F \bigcirc) \Rightarrow (F G \neg \bigcirc)]$$

any path that visits $\bigcirc$

visits $\bigcirc$ finitely many times
Fragments of CTL*

- CTL: each temporal modality is in the immediate scope of a path quantifier.

**Theorem**

*CTL model checking is PTIME-complete.*

*CTL symbolic model checking is PSPACE-complete.*

- LTL: $E\varphi$ or $A\varphi$, where $\varphi$ has no path quantifier.

**Theorem**

*LTL (symbolic) model checking is PSPACE-complete.*
Fragments of CTL*

- CTL: each temporal modality is in the immediate scope of a path quantifier.

**Theorem**

*CTL model checking is PTIME-complete.*

*CTL symbolic model checking is PSPACE-complete.*

- LTL: $E\varphi$ or $A\varphi$, where $\varphi$ has no path quantifier.

**Theorem**

*LTL (symbolic) model checking is PSPACE-complete.*

**Theorem**

*CTL* $(symbolic)$ model checking is PSPACE-complete.
Outline of the presentation

1. Introduction
   - formal verification model checking and synthesis

2. Classical temporal logics: CTL and LTL
   - expressing properties of “closed” systems

3. Temporal logics for games: ATL
   - expressing properties of interacting systems
   - extensions to non-zero-sum games
Reasoning about multi-agent systems

**Concurrent games**

A **concurrent game** is made of

- a transition system;

![Diagram](image.png)
Reasoning about multi-agent systems

Concurrent games

A concurrent game is made of
- a transition system;
- a set of agents (or players);
Reasoning about multi-agent systems

Concurrent games

A concurrent game is made of
- a transition system;
- a set of agents (or players);
- a table indicating the transition to be taken given the actions of the players.
Reasoning about multi-agent systems

Concurrent games

A concurrent game is made of
- a transition system;
- a set of agents (or players);
- a table indicating the transition to be taken given the actions of the players.

Turn-based games

A turn-based game is a game where only one agent plays at a time.
Reasoning about open systems

**Strategies**

A *strategy* for a given player is a function telling what to play depending on what has happened previously.
Reasoning about open systems

**Strategies**

A *strategy* for a given player is a function telling what to play depending on what has happened previously.

Strategy for player □:
alternately go to ⬝ and ⬞.
Reasoning about open systems

**Strategies**

A *strategy* for a given player is a function telling what to play depending on what has happened previously.

Strategy for player □:
alternately go to ● and ●.
A strategy for a given player is a function telling what to play depending on what has happened previously.

Strategy for player □: alternately go to ● and ○.
Reasoning about open systems

**Strategies**

A strategy for a given player is a function telling what to play depending on what has happened previously.

Strategy for player □: alternately go to □ and □.
Reasoning about open systems

Strategies

A strategy for a given player is a function telling what to play depending on what has happened previously.

Strategy for player □:
alternately go to ○ and ●.
Reasoning about open systems

**Strategies**

A *strategy* for a given player is a function telling what to play depending on what has happened previously.

Strategy for player □:
alternately go to ● and ○.
Reasoning about open systems

Strategies

A strategy for a given player is a function telling what to play depending on what has happened previously.

Strategy for player □: alternately go to ◦ and ◦.
Reasoning about open systems

Strategies

A strategy for a given player is a function telling what to play depending on what has happened previously.

Strategy for player □:
alternately go to □ and □.
Reasoning about open systems

**Strategies**

A strategy for a given player is a function telling what to play depending on what has happened previously.

Strategy for player □:
alternately go to □ and □.
Alternating-time Temporal Logic

ATL formulas are built inductively using atomic propositions, Boolean combinations, temporal modalities $\mathbf{X}$ and $\mathbf{U}$, and strategy quantifiers:

$\langle\langle A \rangle\rangle \varphi$ expresses that $A$ has a strategy to enforce $\varphi$. 
Alternating-time Temporal Logic

ATL formulas are built inductively using atomic propositions, Boolean combinations, temporal modalities $X$ and $U$, and strategy quantifiers:

$\langle A \rangle \varphi$ expresses that $A$ has a strategy to enforce $\varphi$. 
Alternating-time Temporal Logic

ATL formulas are built inductively using atomic propositions, Boolean combinations, temporal modalities $X$ and $U$, and strategy quantifiers:

$\langle A \rangle \varphi$ expresses that $A$ has a strategy to enforce $\varphi$. 
Alternating-time Temporal Logic

ATL formulas are built inductively using atomic propositions, Boolean combinations, temporal modalities $X$ and $U$, and strategy quantifiers:

\[ \langle A \rangle \varphi \text{ expresses that } A \text{ has a strategy to enforce } \varphi. \]
Alternating-time Temporal Logic

ATL formulas are built inductively using atomic propositions, Boolean combinations, temporal modalities $X$ and $U$, and strategy quantifiers:

$\langle A \rangle \varphi$ expresses that $A$ has a strategy to enforce $\varphi$. 
Alternating-time Temporal Logic

ATL formulas are built inductively using atomic propositions, Boolean combinations, temporal modalities $X$ and $U$, and strategy quantifiers:

$$\langle A \rangle \varphi$$ expresses that $A$ has a strategy to enforce $\varphi$. 

![Diagram of ATL logic formulas]

- $\langle \bigcirc \rangle F$  
- $\langle \Box \rangle F$  
- $\langle \bigcirc \rangle G(\langle \Box \rangle F)$
Alternating-time Temporal Logic

ATL formulas are built inductively using atomic propositions, Boolean combinations, temporal modalities $\mathbf{X}$ and $\mathbf{U}$, and strategy quantifiers:

$$\langle \langle \mathbf{A} \rangle \rangle \varphi$$ expresses that $A$ has a strategy to enforce $\varphi$. 

\[ 
\begin{align*}
\langle \langle \mathbf{A} \rangle \rangle \varphi & \equiv \langle \langle \mathbf{G} \langle \langle \mathbf{F} \rangle \rangle \mathbf{p} \rangle \rangle \\
& \equiv \langle \langle \mathbf{G} \langle \langle \mathbf{p} \rangle \rangle \mathbf{p} \rangle \rangle
\end{align*} \]
Alternating-time Temporal Logic

ATL formulas are built inductively using atomic propositions, Boolean combinations, temporal modalities $X$ and $U$, and strategy quantifiers:

$\langle A \rangle \varphi$ expresses that $A$ has a strategy to enforce $\varphi$. 
Alternating-time Temporal Logic

ATL formulas are built inductively using atomic propositions, Boolean combinations, temporal modalities $X$ and $U$, and strategy quantifiers:

\[ \langle A \rangle \varphi \] expresses that $A$ has a strategy to enforce $\varphi$.

**Theorem**

*ATL model checking is PTIME-complete.*

*ATL symbolic model checking is EXPTIME-complete.*
Another semantics: ATL with strategy contexts

\[ G(\Diamond) \]
Another semantics: ATL with strategy contexts

\[ ⟨⟨G⟩⟩F \]

consider the following strategy of Player \( \bigcirc \): “always go to \( \square \)”;
Another semantics: ATL with strategy contexts

Consider the following strategy of Player \( \bigcirc \): “always go to \( \square \)”;
Another semantics: ATL with strategy contexts

consider the following strategy of Player $\bigcirc$: “always go to $\square$”; in the remaining tree, Player $\square$ can always enforce a visit to $\bigcirc$. 

$⟨⟨\bigcirc⟩⟩ G (⟨⟨\square⟩⟩ F \bigcirc)$
ATL with strategy contexts

Definition

\(\text{ATL}_{sc}\) has two new strategy quantifiers: \(\langle \cdot A \rangle \varphi\) and \(\parallel A \parallel \varphi\).

- \(\langle \cdot A \rangle\) is similar to \(\langle A \rangle\) but assigns the corresponding strategy to \(A\) for evaluating \(\varphi\);
- \(\parallel A \parallel\) drops the assigned strategies for \(A\).
ATL with strategy contexts

Theorem

$ATL_{sc}$ is strictly more expressive than ATL.
**Theorem**

$\text{ATL}_{sc}$ *is strictly more expressive than* $\text{ATL}$.

**Proof**

$$\langle A \rangle \varphi \equiv \langle \text{Agt} \rangle \langle \cdot A \cdot \rangle \hat{\varphi}$$
Theorem

\( \text{ATL}_{sc} \) is strictly more expressive than ATL.

Proof

\[ \langle 1 \rangle (\langle 2 \rangle \mathbf{X} a \land \langle 2 \rangle \mathbf{X} b) \] is only true in the second game. But ATL cannot distinguish between these two games.
What $\text{ATL}_{sc}$ can express

- All $\text{ATL}^*$ properties;
What $\text{ATL}_{sc}$ can express

- All $\text{ATL}^*$ properties;
- Client-server interactions for accessing a shared resource:

\[
\langle \text{Server} \rangle \ G \wedge \left[ \bigwedge_{c \in \text{Clients}} \langle \cdot \rangle F \text{access}_c \right]
\]

Existence of Nash equilibria:

\[
\langle \cdot \rangle A_1, \ldots, A_n \wedge \bigwedge i \left( \langle \cdot \rangle A_i \phi_A_i \Rightarrow \phi_A_i \right)
\]

Existence of dominating strategy:

\[
\langle \cdot \rangle A \wedge \left[ \langle \cdot \rangle B \neg \phi \Rightarrow \langle \cdot \rangle A \neg \phi \right]
\]
What $\text{ATL}_{sc}$ can express

- All $\text{ATL}^*$ properties;
- Client-server interactions for accessing a shared resource:
  \[
  \langle \cdot \text{Server} \cdot \rangle \ \mathbf{G} \ \bigg[ \bigwedge_{c \in \text{Clients}} \langle \cdot c \cdot \rangle \ \mathbf{F} \ \text{access}_c \bigg] 
  \wedge 
  \neg \bigwedge_{c \neq c'} \text{access}_c \wedge \text{access}_{c'}
  \bigg]\]

- Existence of Nash equilibria:
  \[
  \langle \cdot A_1, \ldots, A_n \cdot \rangle \ \bigwedge_i \ \big( \langle \cdot A_i \cdot \rangle \ \varphi_{A_i} \ \Rightarrow \ \varphi_{A_i} \big) 
  \]
What $\text{ATL}_{sc}$ can express

- All $\text{ATL}^*$ properties;

- **Client-server interactions** for accessing a shared resource:

\[
\langle \text{Server} \rangle \ G \left[ \bigwedge_{c \in \text{Clients}} \langle \cdot \rangle F \text{access}_c \right. \\
\left. \bigwedge \neg \bigwedge_{c \neq c'} \text{access}_c \land \text{access}_{c'} \right]
\]

- Existence of **Nash equilibria**:

\[
\langle \cdot \rangle A_1, \ldots, A_n \rangle \bigwedge_i (\langle \cdot \rangle A_i \varphi_{A_i} \Rightarrow \varphi_{A_i})
\]

- Existence of **dominating strategy**:

\[
\langle \cdot \rangle A \ [B] (\neg \varphi \Rightarrow [A] \neg \varphi)
\]
Model checking $\text{ATL}_{\text{sc}}$

**Theorem**

Given a CGS $\mathcal{C}$, a state $\ell_0$ and an $\text{ATL}_{\text{sc}}$ formula $\varphi$, we can build an alternating parity tree automaton $A$ s.t.

$$\mathcal{L}(A) \neq \emptyset \iff \mathcal{C}, \ell_0 \models \emptyset \varphi.$$  

$A$ has size $d$-exponential, where $d$ is the maximal number of nested quantifiers.
Model checking $\text{ATL}_{\text{sc}}$

**Theorem**

Given a CGS $\mathcal{C}$, a state $\ell_0$ and an $\text{ATL}_{\text{sc}}$ formula $\varphi$, we can build an alternating parity tree automaton $\mathcal{A}$ s.t.

$$\mathcal{L}(\mathcal{A}) \neq \emptyset \iff \mathcal{C}, \ell_0 \models \emptyset \varphi.$$  

$\mathcal{A}$ has size $d$-exponential, where $d$ is the maximal number of nested quantifiers.

**Theorem**

Model checking $\text{ATL}_{\text{sc}}$ is $d$-EXPTIME-complete.
Model checking $\text{ATL}_{sc}$

Tree-automata approach

The unwinding tree is accepted by a deterministic tree automaton;
The unwinding tree is accepted by a deterministic tree automaton;
Model checking $\text{ATL}_{sc}$

Tree-automata approach

- A strategy is encoded as a labelling of the unwinding tree;
Model checking $\text{ATL}_{sc}$

Tree-automata approach

We can mark outcomes corresponding to selected strategies;
Model checking $\text{ATL}_{sc}$

Tree-automata approach

We mark the tree with extra propositions $p_l$ and $p_r$, and require that it satisfies $A(G p_o \Rightarrow p_l U p_r)$;
Model checking $\text{ATL}_{sc}$

Tree-automata approach

- We require that subtrees rooted at a $p_l$ or $p_r$ node is accepted by the automaton for $\varphi$ or $\varphi'$, respectively;
Model checking $\text{ATL}_{sc}$

We can build a tree automaton accepting all trees that can be labelled with correct strategies. This requires turning the alternating tree automaton into a non-deterministic one, which yields an exponential-size automaton.
Conclusions

- Our results on $\text{ATL}_{sc}$:
  - $\text{ATL}_{sc}$ is a **natural semantic extension** of the popular ATL;
  - $\text{ATL}_{sc}$ is **much more expressive**: equilibria, client-server interactions... Well-suited for non-zero-sum objectives;
  - There is a price for this expressiveness: **high complexity** of the model-checking algorithm.

Future works:
- study satisfiability of $\text{ATL}_{sc}$;
- behavioural equivalence for $\text{ATL}_{sc}$;
- handle stochastic strategies, partial observation, ...
Conclusions

- **Our results on $\text{ATL}_{sc}$:**
  - $\text{ATL}_{sc}$ is a natural semantic extension of the popular $\text{ATL}$;
  - $\text{ATL}_{sc}$ is much more expressive: equilibria, client-server interactions... Well-suited for non-zero-sum objectives;
  - There is a price for this expressiveness: high complexity of the model-checking algorithm.

- **Future works:**
  - study satisfiability of $\text{ATL}_{sc}$;
  - behavioural equivalence for $\text{ATL}_{sc}$.
  - handle stochastic strategies, partial observation, ...